Part 7: Trust Region Optimization

January 18, 2021

Hey everyone! Today's post will go over something that we have discussed in some detail before: optimization! Specifically, Trust-Region methods for convex optimization. For this we will by happenstance go over Gauss Newton method as well. First let's do the easiest version of gradient based optimization: Gradient Descent. We start with the *Update Rule*: $x_{k+1} = x_k + g$ where in the case of Gradient Descent:

$$g = -\alpha \nabla f(x_k)$$

The descent direction g is equal to the gradient at the point x_k.

Initialize x_0, $\alpha > 0$;
for $k = 1 : K$ **do**
    $x_{k+1} = x_k + \alpha g$;
end

Algorithm 1: General Descent Method

Note that α (step size) is a constant that modulates the size of the difference between x_{k+1} and x_k. Also note that -1 is multiplied by g so that when $g > 0$ (i.e. we go up) Gradient Descent moves in the opposite direction.

1 Newton/Gauss-Newton Method

Next we will talk about the Newton method, which is similar to Gradient Descent but utilizes Hessian $\nabla^2 f(x)$ information:

$$g = -\nabla^2 f(x_k)^{-1} \nabla f(x_k)$$

This leads to the beloved Quasi-Newton methods where $B_k \approx \nabla^2 f(x_k)$ because the Hessian is often difficult to calculate. A modification of Newton's method is Gauss-Newton's method where:

$$g = (\nabla f(x_k)\nabla^T f(x_k))^{-1}\nabla f(x_k) f(x)$$

2 Trust Region Concept

The basic idea behind Trust Regions is that we optimize a function in a small space where we have a good approximation of the surface $f(x)$ through a Taylor Series approximation m around x plus some vector p for gradient g and Hessian approximation B.

$$f(x+p) \approx m(p) = f_k + g_k + 1/2 g_k^T B_k p_k$$

We solve the problem:

$$p^* = argminm(p) \text{ s.t. } ||p|| \leq \delta_k$$

Which basically means minimize m approximation of $f(x)$ for some distance p constrained on the ball of radius δ_k (using the $L2$ norm). The simplest solution to this problem is the Full Step:

$$p_B = -B_k^{-1} g_k \tag{1}$$

Which is optimal when (i) B_k is PD and (ii) $||B_k^{-1} g_k|| <= \delta_k$, which is the unconstrained minimization of $m(p)$.

3 Trust Region Algorithm

We define an improvement metric:

$$\rho_k = \frac{f(x_k) - f(x_k + p_k)}{m_k(0) - m_k(p_k)}$$

Which is the ratio of the reduction in $f(x_k)$ to the approximation in $m(p_k)$. The closer to $\rho_k = 1$ the better. The Trust Region algorithm incorporates this calculation as follows:

Initialize $\hat{\Delta} > 0$ $\Delta_o \in (0, \hat{\Delta})$ $\eta \in [0, 1/4)$;
for $k = 1 : K$ **do**
  Calculate p_k ;
  Calculate ρ_k ;
  if $\rho_k < 1/4$ **then**
    $\Delta_{k+1} = 1/2\Delta_k$;
  else
    if $\rho_k > 3/4$ $||p_k|| = \Delta_k$ **then**
      $\Delta_k = min\{2\Delta_k, \hat{\Delta}\}$
    else
      $\Delta_k = \hat{\Delta}$
  if $\rho_k > \eta$ **then**
    $x_{k+1} = x_k + p_k$
  else
    $x_{k+1} = x_k$
end

Algorithm 2: General Trust Region Method

What we need to do is basically solve for p_k and adjust Δ_k.

3.1 Cauchy Point

The next most basic solution to the subproblem in the Trust Region algorithm is the Cauchy Point.

$$\tau_k = \begin{cases} 1 & \text{if } g_k^T B_k g_k \leq 0 \\ min\{||g_k||^3/(\Delta_k g_k^T B_k g_k), 1\} & \text{otherwise} \end{cases}$$

$$p_c = -\frac{\tau_k \Delta_k g_k}{||g_k||} \tag{2}$$

Think about the Cauchy Point as p which gives the steepest descent subject the the constraint talked about above. One way to implement the Cauchy Point is to use p_B when B_k is PD and $||p_B|| < \Delta_k$, and p_c elsewhere.

3.2 Dogleg Method

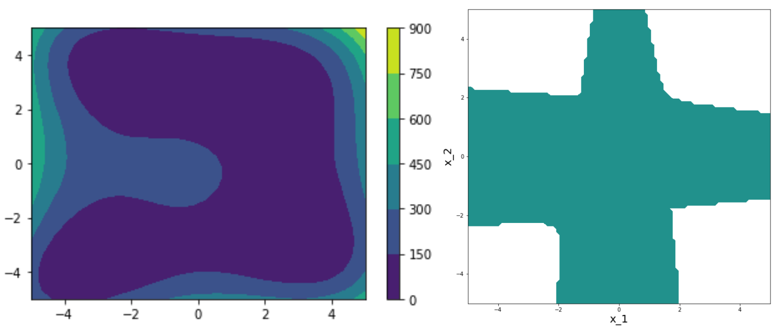
This one took a while to implement but here we go. Specifically we implement the Dogleg Method from Powell's Numerical Optimization Book, which tends to be useful when B_k is PD (but don't take my word for it). After we calculate τ, p_u (steepest descent direction) and p_B (full step solution) we can use the following:

$$p_k = \begin{cases} \tau p_u & \text{if } 0 \leq \tau \leq 1 \\ p_u + (\tau - 1)(p_B - p_u) & \text{if } 1 \leq \tau \leq 2 \end{cases} \tag{3}$$

4 Example Problem

Here we throw some of these methods at the modified 2-D Himmelblau problem:

$$f(x) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2$$

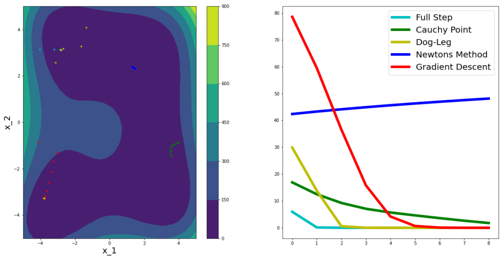


(a) $f(x)$ (b) Locations of non-PD

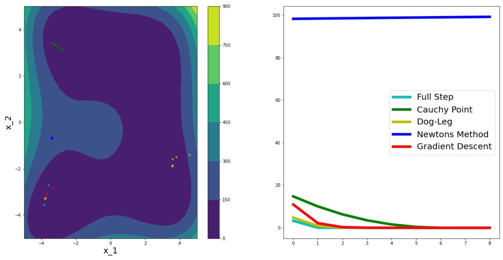
Figure 1: Himmelblau Function

Here we do six random initializations of some of the methods we have talked about.

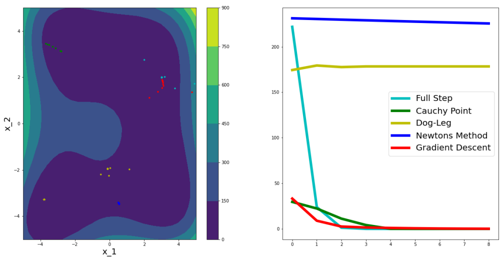
This is a very difficult optimization problem to solve and we did an okay job. In theory we should be taking a look at the positive definiteness of the region we are in and select the method based on that (which is what more sophisticated methods do). Great job everyone! I am not sure what I will present next week so I'll just say we'll do something related to BOTorch.



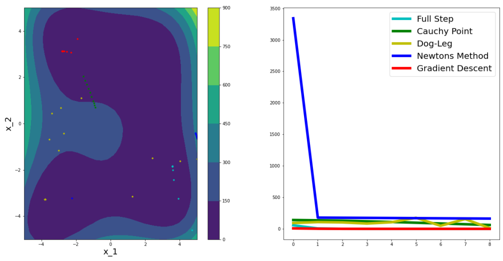
(a)



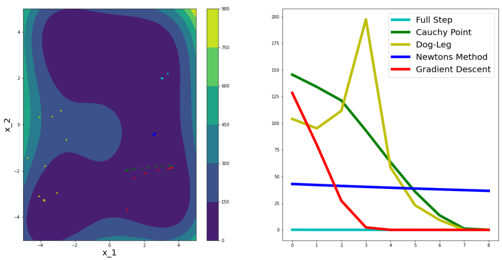
(b)



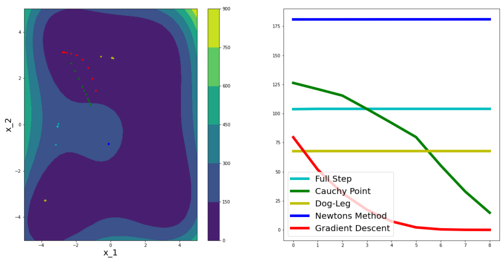
(c)



(d)



(e)



(f)

Figure 2: Example Optimization Problem | $\eta = 0.1$, $\hat{\Delta} = 1$